APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: CONTROLLING DATA FLOW BETWEEN PROCESSOR

SYSTEMS

INVENTOR: RAY M. RICHARDSON

Express Mail No. EL911616518US

Date: SEPTEMBER 27, 2001

CONTROLLING DATA FLOW BETWEEN PROCESSOR SYSTEMS

Background

This invention relates generally to processor-based systems and particularly to systems including two separate

5 processor systems that communicate with one another.

In many wireless systems, a baseband processor is available to handle communication tasks. A multimedia processor is generally available for the wealth of non-communication-based tasks. For example, in cellular

10 telephones, the baseband processor may be responsible for implementing the relevant wireless protocol. Conversely, the multimedia processor may be responsible for controlling the display, providing games, and implementing address book and calendar features and the like.

15 Thus, it is convenient in many wireless systems to provide two processors that operate as intercommunicating systems. That is, each processor system communicates with the other processor system. The processor systems may be separately integrated or commonly integrated on the same

20 chip.

Direct memory access or DMA forms a second data channel between peripherals and main memory through which a peripheral can directly access the main memory without the

1

help of the processor to read or write data. DMA may be implemented by a DMA controller.

Existing DMA controllers are primarily concerned with the internal data flows of a particular process or

5   processor. Streaming data flows between different processors in the same processor-based system add additional complexities that may lead to flow bottlenecks and inefficient use of processor resources. Each DMA controller, in a multi-processor system, may be focused on

10   its associated processor, resulting in too many interrupts to each processor.

Thus, there is a need, in multi-processor systems, to facilitate DMA operations.

Brief Description of the Drawings

15   Figure 1 is a block depiction of a system in accordance with one embodiment of the present invention;

Figure 2 is a flow chart for "send" software in accordance with one embodiment of the present invention; and

20   Figure 3 is a flowchart for "receive" software in accordance with one embodiment of the present invention.

Detailed Description

Referring to Figure 1, a processor-based system 10 may include a pair of processor systems 12 and 14. In one

25   embodiment, the system 10 is a wireless communication

2

system, such as a cellular telephone. The systems 12 and 14 communicate over a bus 13. In one embodiment, the system 12 may be a multimedia processor system and the system 14 may be a baseband processor system. The systems

5   12 and 14 may be integrated on separate or the same integrated circuit.

The system 12 may include a first-in-first-out (FIFO) buffer 18a that is coupled to a direct memory access (DMA) controller 16a that includes a storage 32a in one

10   embodiment. The controller 16a communicates with a linked list of descriptors, indicated as descriptors 26a, 28a, and 30a. Each descriptor 26a, 28a, and 30a is coupled to its respective buffer 20a, 22a, and 24a. The descriptors 26-30 include flags that indicate whether the associated buffer

15   is either empty or full. In the illustrated embodiment, the buffers 20a, 22a, and 24a are illustrated as being in their empty state following a transfer to the system 14, for example.

Similarly, the system 14 includes a first-in-first-out

20   (FIFO) buffer 18b, a controller 16b with a storage 32b in one embodiment. The descriptors 26b, 28b, and 30b are arranged in a linked list, and coupled to associated buffers 20b, 22b, and 24b.

Through the use of the buffers 20-24 and descriptors

25   26-30, inter-processor data flow may be made more efficient in some embodiments. Each of the buffers 20-24 are

3

maintained as a linked list with descriptors 26-30 acting as queue flags to indicate whether the associated buffer 20-24 is either empty or full. This enables software on each system 12 or 14 to freely interact with any of the

5   buffers 20-24.

As shown in Figure 1, immediately following a data transfer, the buffers 20a-24a are designated by descriptors 26a-30a as being empty while the buffers 20b-24b are indicated by their descriptors 26b-30b as being full.

10   Turning to Figure 2, the software 34, that may, for example, be stored in the storage 32a and 32b, may send information across the bus 13 from the system 12 to the system 14, in one example. If both systems 12 and 14 are aware of an impending data transfer, the buffers 20-24 on

15   each side of the interface 13 are prepared as indicated in block 36. The buffers 20a-24b are set with the first data to send as indicated in block 38. The descriptors 26-30 for each linked buffer 20a-24b are prepared in linked list fashion as indicated in block 40. Then, the empty bit is

20   set for each buffer, as indicated in block 42. When ready, DMA requests on both sides are initiated by the corresponding FIFOs 18, as indicated in block 44. The DMA transfer then begins, as indicated in block 46. Data may stream from the buffers 20a-24a through the interface 13

25   and the FIFO 18b to the buffers 20b-24b on the system 14.

4

When data transfer from one source buffer is complete, as determined at diamond 48, the DMA controller 16a sets the empty bit in the corresponding descriptor, as indicated in block 50. The controller 16a then writes the descriptor back to memory, as indicated in block 52, and moves on to the next descriptor in the linked list as indicated in block 54. Before transferring the data from a buffer, the controller 16a checks the empty bit, as indicated in diamond 56. If the empty bit is set, the controller 16a causes an interrupt, as indicated in block 58. Software intercepts this interrupt, fills the buffers 20a-24a with more data, clears the empty bit in each descriptor 26a-30a and starts the DMA channel again by setting a run bit.

As shown in Figure 3, the receive software 60 prepares the buffers 20b-24b in the system 14, as indicated in block 62. The software 60 may be stored in storage 32. The descriptors are prepared, as indicated in block 64, the full bit is clear as indicated in block 66 and the DMA channels are prepared to receive the data. When ready, the DMA requests are initiated by the FIFOs 18, as indicated in the block 68. The DMA transfer then proceeds, as indicated in block 70, with data streaming from memory buffers in one processor system (12 or 14) to the other processor system (12 or 14).

When the data transfer from the source buffers (in this case the buffers 20a-24a) is complete, a check at

diamond 72 determines when a target buffer is full. The controller 16b sets the full bit, as indicated in block 74 in the corresponding descriptor 26b-30b, writes the descriptor back to memory, as indicated in block 76, and

5   moves on to the next descriptor in the linked list, as indicated in block 78. Before the controller 16b attempts to fill the next buffer, it checks the full bit, as indicated in diamond 80. If the full bit is set, the controller 16b generates an interrupt, as shown in block

10   82. Software intercepts this interrupt, copies the buffers into other locations, clears the full bit in each descriptor and starts the DMA channel again by setting the run bit.

     In some embodiments, the software is able to detect

15   empty and full buffers and, even as DMA transfer continues, perform the necessary handling before an interrupt becomes necessary. In this manner, the number of interrupts may be greatly reduced. Thus, source buffers may be refilled and target buffers may be emptied to continue data transfer.

20   In one embodiment, the empty and full flags may be fully interchangeable. In such an embodiment, the same flag may be used to indicate "empty" when the DMA buffer descriptor is used to transmit data and "full" when the DMA buffer descriptor is used to receive data.

25   While the present invention has been described with respect to a limited number of embodiments, those skilled

in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

5   What is claimed is: